Inferring connectivity from imaging activity

Joshua Vogelstein, some others, Liam Paninski

December 17, 2008

1 Introduction

Our goal is to estimate the connection matrix of a population of neurons from simultaneously imaging their activity.

2 Model

2.1 Single neuron model

We assume that we have a discrete time model, with time step size Δ and T total time steps. In each time step, the neuron can emit any non-negative number of spikes, i.e., $n_t \in \{0, 1, 2, \cdots\} = \mathbb{N}_0$. The spiking is governed by a Poisson process with rate $\lambda_t \Delta$, where λ_t is given by:

$$\lambda_t = \exp\{b + \boldsymbol{k}'\boldsymbol{s}_t + \omega h_t\} \tag{1}$$

$$h_t - h_{t-1} = -\frac{\Delta}{\tau_h} h_t + n_{t-1} + \sigma_h \sqrt{\Delta}\varepsilon \tag{2}$$

calcium model:

$$[\text{Ca}^{2+}]_t - [\text{Ca}^{2+}]_{t-1} = -\frac{\Delta}{\tau_c}([\text{Ca}^{2+}]_t + [\text{Ca}^{2+}]_b) + A n_t + \sigma_c \sqrt{\Delta}\varepsilon, \tag{3}$$

observation model:

$$F_t = \alpha S([\text{Ca}^{2+}]_t) + \beta + (S([\text{Ca}^{2+}]_t) + \sigma_F)\varepsilon_t \tag{4}$$

where $S(x) = x^n/(x^n + k_d)$ is the standard Hill equation, the ε's indicate standard normal random variables.

2.2 Multiple neuron model

We assume the firing rate for neuron i, $\lambda_{i,t}\Delta$, is given by:

$$\lambda_{i,t} = f(b_i + \boldsymbol{k}_i'\boldsymbol{s}_t + \sum_{j=1}^{N} \omega_{ij} h_{j,t}) \tag{5}$$

$$h_{i,t} - h_{i,t-1} = -\frac{\Delta}{\tau_{h_i}} h_{i,t} + n_{i,t-1} + \sigma_{h_i}\sqrt{\Delta}\varepsilon, \tag{6}$$

where each h_i corresponds to the spike history term associated with neuron i, and has an associated time-constant, τ_i. The input each neuron recieves, therefore, is the sum of spike history terms from all neurons (including itself). Thus, $\boldsymbol{\omega} = \{\omega_{ij}\} = \{\omega_{11}, \omega_{12}, \ldots, \omega_{1N}, \ldots, \omega_{NN}\}$ comprises the weight matrix for these neurons.

calcium model:

$$[\text{Ca}^{2+}]_{i,t} - [\text{Ca}^{2+}]_{i,t-1} = -\frac{\Delta}{\tau_{c_i}}([\text{Ca}^{2+}]_{i,t} + [\text{Ca}^{2+}]_{b_i}) + A_i n_{i,t} + \sigma_{c_i}\sqrt{\Delta}\varepsilon, \tag{7}$$

observation model:

$$F_{i,t} = \alpha_i S([\text{Ca}^{2+}]_{i,t}) + \beta_i + (S([\text{Ca}^{2+}]_{i,t}) + \sigma_{F_i})\varepsilon, \tag{8}$$

Note that we assume that $S(\cdot)$ is the same for each neuron (i.e., the nonlinear function is the same for each neuron, because the parameters are a function of the indicator, not the neuron). Further note that we assume (for now) that the noise on the calcium and observation for each neuron is independent (an assumption that we will probably want to relax later).

2.3 Mathematical Background

To estimate $\boldsymbol{\omega}$, we adopt a state-space framework. More specifically, let y_t be the observations, and x_t be the *hidden states*. We then have:

$$\boldsymbol{y}_t \sim g_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{x}_t) \tag{9}$$

$$\boldsymbol{x}_t \sim f_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{10}$$

where $g_{\boldsymbol{\theta}}(\cdot)$ and $f_{\boldsymbol{\theta}}(\cdot)$ indicate the *observation* and *transition* distributions, respectively (and we have adopted the shorthand notation $f_{\boldsymbol{\theta}}(\cdot) = f(\cdot|\boldsymbol{\theta})$. Letting $\boldsymbol{X}_t = \{X_{i,t}\} = \{X_{1,t}, \ldots, X_{N,t}\}$, for the above model, we have $\boldsymbol{y}_t = \boldsymbol{F}_t$, $\boldsymbol{x}_t = \{\boldsymbol{n}_t, \boldsymbol{h}_t, [\textbf{Ca}^{2+}]_t\}$, and $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\tau_c}, [\text{Ca}^{\mathbf{2+}}]_{\boldsymbol{b}}, \boldsymbol{\sigma_c}, \boldsymbol{b}, \{\boldsymbol{k}_i\}, \boldsymbol{\omega}, \boldsymbol{\tau_h}, \boldsymbol{\sigma}_h\}$. Note that given the above model, $\boldsymbol{y}_t \in \Re_+^N$, i.e., each y_t is a non-negative real number, and $\boldsymbol{x}_t \in \Re_+^{2N} \times \{0,1\}^N$, i.e., $\boldsymbol{x}_t$ lies in a $3N$-dimensional space, where $2N$ dimensions are non-negative real numbers ($[\text{Ca}^{2+}]_t$ and h_t for each neuron), and the rest are binary ($n_t \in \{0,1\}$ for each neuron). Thus, we have a "hybrid" system, in that some of our hidden states are continuous, and others are discrete. We may now formally state our goal as:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, P_{\boldsymbol{\theta}}(\boldsymbol{y}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \int P_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{x}) d\boldsymbol{x} \tag{11}$$

where, we have introduced the notation, $\boldsymbol{y} = \boldsymbol{y}_{0:T}$, and $P_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{x})$ may be factored as:

$$P_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{x}) = \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_0) \prod_{t=1}^{T} g_{\boldsymbol{\theta}}(\boldsymbol{y}_t | \boldsymbol{x}_t) f_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}) \tag{12}$$

where $\pi_{\boldsymbol{\theta}}(\boldsymbol{x}_0)$ is the initial distribution of $\boldsymbol{x}$. Sadly, for our model, integral in (11) is computationally intractable. Therefore, we adopt an EM approach:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, Q(\boldsymbol{\theta}, \boldsymbol{\theta}') \tag{13}$$

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}') = E_{p_{\boldsymbol{\theta}'}(\boldsymbol{x}|\boldsymbol{y})} \ln p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{x}) = \int p_{\boldsymbol{\theta}'}(\boldsymbol{x}|\boldsymbol{y}) \ln p_{\boldsymbol{\theta}}(\boldsymbol{y}, \boldsymbol{x}) d\boldsymbol{x} \tag{14}$$

Because we have a state-space model, the above integral may be simplified:

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}') = \int p_{\boldsymbol{\theta}'}(\boldsymbol{x}_0 | \boldsymbol{y}) \times \ln \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_0) d\boldsymbol{x}_0 + \\ \sum_{t=1}^{T} \iint p_{\boldsymbol{\theta}'}(\boldsymbol{x}_t, \boldsymbol{x}_{t-1} | \boldsymbol{y}) \times \ln f_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}) d\boldsymbol{x}_t d\boldsymbol{x}_{t-1} + \\ \sum_{t=0}^{T} \int p_{\boldsymbol{\theta}'}(\boldsymbol{x}_t | \boldsymbol{y}) \times \ln g_{\boldsymbol{\theta}}(\boldsymbol{y}_t | \boldsymbol{x}_t) d\boldsymbol{x}_t \end{aligned} \tag{15}$$

As we have a nonlinear observation model, and nonlinear transition model, the integrals in (15) are intractable as well, so they must be approximated. We adopt a monte carlo strategy, in which we approximate the above integrals with sums. This requires approximating the continuous valued hidden states with discrete valued hidden states. The key is to be able to estimate the pairwise joint conditionals:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{x}_{t-1} | \boldsymbol{y}) \approx \sum_{k,m=1}^{k} \widetilde{p}_{\boldsymbol{\theta}}\big(\boldsymbol{x}_t^{(k)}, \boldsymbol{x}_{t-1}^{(m)} | \boldsymbol{y}\big) \delta_{\boldsymbol{x}_t^{(k)}}(\boldsymbol{x}_t) \delta_{\boldsymbol{x}_{t-1}^{(m)}}(\boldsymbol{x}_{t-1}) \tag{16}$$

where $\delta_{\boldsymbol{x}}(\cdot)$ denotes the Dirac mass at point $\boldsymbol{x}$, and $\widetilde{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_t^{(k)}, \boldsymbol{x}_{t-1}^{(m)} | \boldsymbol{y})$ is computed pointwise for each $\boldsymbol{x}_t^{(k)}$ and $\boldsymbol{x}_{t-1}^{(m)}$, and then normalized such that $\sum_{k,m=1}^{k} \widetilde{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_t^{(l)} | \boldsymbol{x}_{t-1}^{(m)}) = 1$. From this approximation, we can also estimate the marginal conditionals and initial distribution, by integrating out $\boldsymbol{x}_{t-1}$. The key is to efficiency discretize the space, so L is small and the approximation is still good. We adopt a Monte Carlo strategy, in which we represent the pairwise joint conditionals as a product of terms:

$$\widehat{P}_{\boldsymbol{\theta}}(F_v|[\text{Ca}^{2+}]_t)\big(\boldsymbol{x}_t^{(l)}, \boldsymbol{x}_{t-1}^{(m)}|\boldsymbol{y}\big) = \widehat{P}_{\boldsymbol{\theta}}(F_v|[\text{Ca}^{2+}]_t)\big(\boldsymbol{x}_t^{(l)}|\boldsymbol{y}\big) \frac{\widetilde{f}_{\boldsymbol{\theta}}\big(\boldsymbol{x}_t^{(l)}|\boldsymbol{x}_{t-1}^{(m)}\big) w_{t-1}^{(m)}}{\sum_{m=1}^{L} \widetilde{f}_{\boldsymbol{\theta}}\big(\boldsymbol{x}_t^{(l)}|\boldsymbol{x}_{t-1}^{(m)}\big) w_{t-1}^{(m)}} \tag{17}$$

where $w_t^{(l)}$ is defined by

$$P_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) = \sum_{l=1}^{L} w_t^{(l)} \delta_{\boldsymbol{x}_{0:t}^{(l)}}(\boldsymbol{x}_{0:t}) \tag{18}$$

The maximize efficiency, therefore, we aim to choose $\boldsymbol{x}_t^{(l)}$'s to make their associated weights, $w_t^{(l)}$'s approximately equal. In the next section, we describe a number of algorithms, with increasing complexity and (hopefully) utility.

Thus, it should be clear from the above that ideally, one would ideally like to sample jointly from $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{y}_t)$. As this is so high dimensional (more specifically, $3N$ dimensions), the importance sampling idea breaks down (as it is incapable of approximating a high dimensional space with a small number of particles). Therefore, we develop a number of approaches to combat this curse of dimensionality.

2.4 population dynamics

plugging model described in section 2.2 into (9) and (10), we have:

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \prod_{i=1}^{N} p([\text{Ca}^{2+}]_{i,t}|[\text{Ca}^{2+}]_{i,t-1}, n_{i,t}) p(n_{i,t}|\boldsymbol{h}_t) p(h_{i,t}|h_{i,t-1}, n_{i,t-1}) \tag{19}$$

$$p(\boldsymbol{y}_t|\boldsymbol{x}_t) = \prod_{i=1}^{N} p(F_{i,t}|[\text{Ca}^{2+}]_{i,t}) \tag{20}$$

where $\boldsymbol{h}_t = h_{1,t}, \ldots, h_{N,t}$ and the above distributions are defined by

$$p([\text{Ca}^{2+}]_{i,t}|[\text{Ca}^{2+}]_{i,t-1}, n_t) = \mathcal{N}([\text{Ca}^{2+}]_{i,t}; a_{c_i}[\text{Ca}^{2+}]_{i,t-1} + b_i + A_i n_{i,t}; \sigma_{c_i}^2 \Delta) \tag{21}$$

$$p(n_{i,t}|\boldsymbol{h}_{i,t}) = \mathcal{B}(n_{i,t}; \lambda_{i,t}) \tag{22}$$

$$p(h_{i,t}|h_{i,t-1}, n_{i,t-1}) = \mathcal{N}(h_{i,t}; a_{h_i} h_{i,t-1} + n_{i,t-1}, \sigma_{h_i}^2 \Delta) \tag{23}$$

$$p(F_{i,t}|[\text{Ca}^{2+}]_{i,t}) = \mathcal{N}(F_{i,t}; \alpha_i S([\text{Ca}^{2+}]_{i,t}) + \beta_i, (S([\text{Ca}^{2+}]_{i,t}) + \sigma_{F_i})^2) \tag{24}$$

and we have introduced $a_{c_i} = 1 - \Delta/\tau_{c_i}$, $b_i = -\Delta[\text{Ca}^{2+}]_{b_i}/\tau_{c_i}$, and $a_{h_i} = 1 - \Delta/\tau_{h_i}$, for brevity.

3 Algorithms

3.1 Default algorithm: see [Vogelstein et al., 2008] for details

Initialization :

1. Manually define an ROI around each neuron.
2. Average pixel intensity from all pixels in a frame to obtain $F_{i,t}$, i.e., the fluorescence intensity indexed by cell indentity i and frame number t.
3. Separately for each neuron, implement PFS as described in [Vogelstein et al., 2008] (ie, with all cross-coupling terms turned off).

Recursion :

1. For each neuron, let the input, $\boldsymbol{u}_{i,t} = [\boldsymbol{s}_t, \{\widetilde{\boldsymbol{h}}_{i,t}\}]$, where $\{\widetilde{\boldsymbol{h}}_{i,t}\} = \{E[\boldsymbol{h}_{i,t}|\boldsymbol{F}]$ for all $i\}$.
2. Use PFS from [Vogelstein et al., 2008] to learn $\boldsymbol{\omega}$

3.2 Population PF

1. same as 3.1
2. same as 3.1
3. particle filter step: this generalizes 3.1 by sampling not just on self-histories, but also cross-histories. because the transition distribution factorizes as above, we can sample from the optimal one observation ahead sampler, ie, $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{y}_t)$ (by approximating the likelihood term as a Gaussian in $[\text{Ca}^{2+}]_{i,t}$). in particular, to sample spikes after integrating out everything, one is left with:

$$n_{i,t} \sim p(n_{i,t}|\boldsymbol{h}_t)\mathcal{N}([\text{Ca}^{2+}]_{i,t}; \mu_{c_{i,t}}, \sigma^2_{c_{i,t}}) \tag{25}$$

   where $\mathcal{N}([\text{Ca}^{2+}]_{i,t}; \mu_{c_{i,t}}, \sigma^2_{c_{i,t}}$ is a Gaussian approximation to the product of the likelihood and calcium update terms.
4. same as 3.1
5. estimate parameters using standard EM stuff as in [Vogelstein et al., 2008]

it's not obvious whether this approach will outperform the approach outlined in 3.1

3.3 Gibbsish Population PF

this generalizes 3.2, by sampling conditioned not just on spike histories, but also on spike futures from other neurons. in particular, we want to sample spikes for one neuron conditioned on its own past spikes, and *all* spikes from all other neurons. having already completely an iteration, we now sample spikes according to:

$$n_{i,t} \sim p(n_{i,t}|\boldsymbol{h}_t, \boldsymbol{h}_{\setminus i,t+1})\mathcal{N}([\text{Ca}^{2+}]_{i,t}; \mu_{c_{i,t}}, \sigma^2_{c_{i,t}}) \tag{26}$$

where $\boldsymbol{h}_{\setminus i,t+1} = \{h_{1,t+1}, \ldots, h_{i-1,t+1}, h_{i+1,t+1}, \ldots, h_{N,t+1}\}$. we compute this using the method described in [Pillow and Latham, 2007]. For future spikes, we let the posterior mean of the spike train from the previous EM iteration correspond to future spikes.

3.4 Gibbser Population PF

this generalizes 3.3, by sampling not just sampling based on previous EM iteration's mean posterior spike train, but actually doing a proper gibbs step within each forward pass. in particular, we do the sampling as described in 3.3 to initialize, and obtain a spike train for each neuron. we then iterate, sampling for each neuron, conditioned on the entire spike trains for other neurons. by gibbs, we will eventually converge to having sampled jointly from them all. i am not convinced that this is a particularly useful generalization of the 3.3.

3.5 Metropolis-Gibbs Population PF

this generalizes 3.4, by placing the Gibbs sampler just described into a MCMC approach. more specifically, we proceed as follows. for each neuron:

1. generate $M-1$ particles, sampling according to $p(n_{i,t}|\boldsymbol{h}_t, \boldsymbol{h}_{\setminus i,t+1}, F_{i,t})$ as described in 3.4.
2. add current path to population of particles and compute appropriately normalized transition probabilities
3. use standard forward-backward sampling algorithm for HMM's to sample $\boldsymbol{x}^*$ from this augmented space
4. compute $q(\boldsymbol{x}^*)$, the probability of sampling z using a forward-backward recursion
5. compute probability of acceptance, $r = \frac{q(\boldsymbol{x})p(\boldsymbol{x}^*)}{q(\boldsymbol{x}^*)p(\boldsymbol{x})}$, where $\boldsymbol{x}$ is the current path and $p(\boldsymbol{x}^*)$ is the posterior

we iterate this step until we accept a new spike train for a particular neuron, and then repeat for all subsequent neurons. alternately, we could randomly choose a neuron after each iteration. i have no idea which approach would be better.

speeding things up as this probably will take a while, we can do a number of things to speed it up

- skip generating particles step sometimes, ie, keep sampling from a particular HMM until prob of acceptance gets too small
- do an increment/decrement thing. in other words, after generating the first set of particles, with each additional acceptance, we simply eliminate a path (based on its likelihood). thus, we must only generate $M-1$ extra particles once.
- instead of using the standard hmm sampling to generate a new proposal, use Viterbi for PF (as described in [Godsill et al., 2001], or the fast version of that approach (as described in [Klaas et al., 2005] to generate a possible new sequence. then, if that one is not accepted, use standard hmm sampling to generate other proposals.

3.6 Ways we might make stuff better

these ideas potentially improve on all the above ideas.

1. Improved observation model — Identify ROI for each neuron using some ImageJ plug-in (or other such method). Instead of equally weighting all pixels, initialize pixel weights with first eigenvector computed using SVD, and then proceed as before. upon constructing SS, update spatial filter weights along with other parameters. An analysis of the kind of spatial filters should facilitate parameterizing the filter (potentially as a mixture of Gaussians). Alternately, an appropriate regularizing kernel may help. Note that if two or more neurons are in the same ROI, the observation model must be modified somewhat (but I'm not putting those details in here at this time)

2. Improved transition model — The most obvious improvement would be to allow additional time constants for the dynamics (either calcium or fluorescence). Perhaps the best way to decide what to include would be to take data for which we have both images and ephys, and fit a couple parametric models, and do cross-validation to compare which is best. The options would include: (i) an additional calcium state, (ii) temporal dynamics for the fluorescence, (iii) no noise on calcium, or (iv) some combination of the above.

3. Rao-blackwellized Particle Filter (RBPF) — This would amount to having a mixture of kalman filters (MKF), each conditioned on a particular spike train. As far as reducing unnecessary variance, I think it would be hard to justify not incorporating this into our model (see [Chen and Liu, 2000] for details). Note, however, that in the limit as $N \to \infty$, the particle approximation is perfect, whereas the RBPF (or MKF) is not, because we must estimate the likelihood as a Gaussian function of [Ca^{2+}].

3.7 these are not very good ideas, but i thought they were potentially worth writing down

1. Improved resampling — we could use a number of tricks here
   - sample $N_p' > N_p$ particles, but only resample using N_p particles, aka, prior boosting (see [GREEN, 1995, Carpenter et al., 1999] for details).
   - Rejection sampling — We do not resample, but rather, only accept particles at each time. See [Chen, 2003] Section E for a summary (and refernces therein for details).
   - robust weighting (to handle outliers) (see review on particle filters)

2. MCMC with particle filtering — there are a number of ways to do this kind of thing
   - RESAMPLE-MOVE —
   - Metropolized Gibbs sampler — see [LIU, 1996] for details

3. data-augmentation for parameter estimation (ie, iterate estimating $p(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{\theta})$ and $p(\boldsymbol{\theta}|\boldsymbol{y},\boldsymbol{x})$ to eventually converge to $p(\boldsymbol{x},\boldsymbol{\theta},\boldsymbol{y})$).

4. use algorithm in continuous-time particle-filter paper ([Ng et al., 2005]) for parameter estimation

References

[Carpenter et al., 1999] Carpenter, J., Clifford, P., and Fearnhead, P. (1999). Improved particle filter for nonlinear problems. *Radar, Sonar and Navigation, IEE Proceedings-*, 146(1):2–7.

[Chen and Liu, 2000] Chen, R. and Liu, J. (2000). Mixture Kalman filters. *Journal of the Royal Statistical Society: Series B (Methodological)*, 62(3):493–508.

[Chen, 2003] Chen, Z. (2003). Bayeian Filtering: From Kalman Filters to Particle Filters, and Beyond.

[Godsill et al., 2001] Godsill, S., Doucet, A., and West, M. (2001). Maximum a Posteriori Sequence Estimation Using Monte Carlo Particle Filters. *Annals of the Institute of Statistical Mathematics*, 53(1):82–96.

[GREEN, 1995] GREEN, P. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732.

[Klaas et al., 2005] Klaas, M., Lang, D., and de Freitas, N. (2005). Fast maximum a posteriori inference in Monte Carlo state spaces. *Artificial Intelligence and Statistics*.

[LIU, 1996] LIU, J. (1996). Peskun's theorem and a modified discrete-state Gibbs sampler. *Biometrika*, 83(3):681–682.

[Ng et al., 2005] Ng, B., Pfeffer, A., and Dearden, R. (2005). Continuous time particle filtering.

[Pillow and Latham, 2007] Pillow, J. and Latham, P. (2007). Neural characterization in partially observed populations of spiking neurons. *Neural Information Processing Systems*.

[Vogelstein et al., 2008] Vogelstein, J., Watson, B., AM, P., Yuste, R., B, J., and Paninski, L. (2008). Spike inference from calcium imaging using sequential monte carlo methods. *Biophysical Journal*.